# Quantum Physics and Jupyter
# Getting Started

January 4, 2017

## Why Programming?

Physicists have traditionally been divided into two (mostly friendly) camps: experimentalists and theorists. However, in parallel with the development of modern computers and programming techniques, a third camp has developed: computationalists. Computational physicists can be found in just about every field of physics, from cosmology to biophysics, and the problems they attempt to solve range from wrangling the enormous data sets produced by CERN to simulating how fires spread through buildings.

Although some physicists specialize in advanced computational methods, it has become increasingly important for physicists of all stripes to learn some programming techniques. It's not often that one encounters a frictionless surface, or a graduate student who wants to spend their nights collecting thousands of voltage readings! Computational methods allow scientists to ease tedious tasks as well as explore ideas well beyond the constraints of a chalkboard.

Luckily, it's really not too hard to pick up some useful programming skills. Throughout this class we will use numerical methods to explore and visualize quantum mechanics, which will hopefully help us understand some of the concepts as well as extend our reach beyond the very limited number of analytically solvable problems. Our programming language of choice will be Python, because it's both easy to learn and very powerful.

## Agenda

This is a tutorial to help you get setup and ready for the first computational exercises. If you're not familiar with Python or Jupyter, you should make sure to complete it before attempting the exercises! Here's what you need to do:

1. Obviously you'll need access to a computer to program! Please tell your instructor *immediately* if you do not have a computer so that they can find a work around.

2. Install Anaconda for your operating system of choice. Pick the Python 3.5 version. Anaconda is an easy to install and use open source, high performance package of everything you'll need for the exercies. In particular, it includes

(a) Python 3.5, as well as the libraries you'll need, like NumPy for math and matplotlib for visualization.

(b) Jupyter, which is a super nifty way of programming in Python that sort of resembles a Mathematica notebook. A Jupyter notebook is a document that you view through your web browser, in which you can run Python code as well as write nice looking text and equations to explain what you're doing. Here is an example of one that let's you "hear" the energy spectrum of hydrogen.

3. Run the example Jupyter notebook, `JupyterTutorial.ipynb`. (`.ipynb` is the file extension for Jupyter Python notebooks.) This notebook will walk you through some of the basics of Python and Jupyter, with fill in the blank sections for you to test your knowledge. Feel free to experiment with this notebook!

4. Once you've completed the exercise at the end of the notebook, please upload the notebook file to Canvas. Again, this will not be graded, but we want to make sure that everyone can get Jupyter up and running.

Note: If you already have Python and Jupyter installed and know what you're doing, you don't have to install Anaconda. For this class you should be able to use either Python 2 or Python 3, though please use Python 3 compliant `print` statements (with parenthesis).

# Details

The installation of Anaconda is usually straightforward, but it's likely that some people will run into issues. Your first response to these issues should be to turn to Google, which will probably direct you to StackOverflow, the center of the debugging universe. If that doesn't help, shoot your instructor a message or go to their office hours!

Once you have Anaconda installed, you'll want to start up the Jupyter notebook. Just how you do this depends on your operating system. On Linux and Mac, you open a terminal and type:

```
> jupyter notebook
```

On Windows, just search for the application `Jupyter Notebook`. You could also open the `Anaconda Prompt` program, which starts up a terminal. In the terminal, you can navigate to a desired directory and start up Jupyter there with the same command as on Linux.

Either way, Jupyter will open up as a new tab in your browser. Here's what it looks like on Windows in Chrome. 1 (No idea how those video games ended up there...)

When Jupyter first starts, you'll see a file directory. On Linux or Max, you'll be in whatever directory you were in your terminal. On Windows, you'll probably be in your home folder, as in the screenshot. In order to open the `JupyterTutorial.ipynb` notebook, just navigate to the folder you saved it in (within the Jupyter browser tab!)

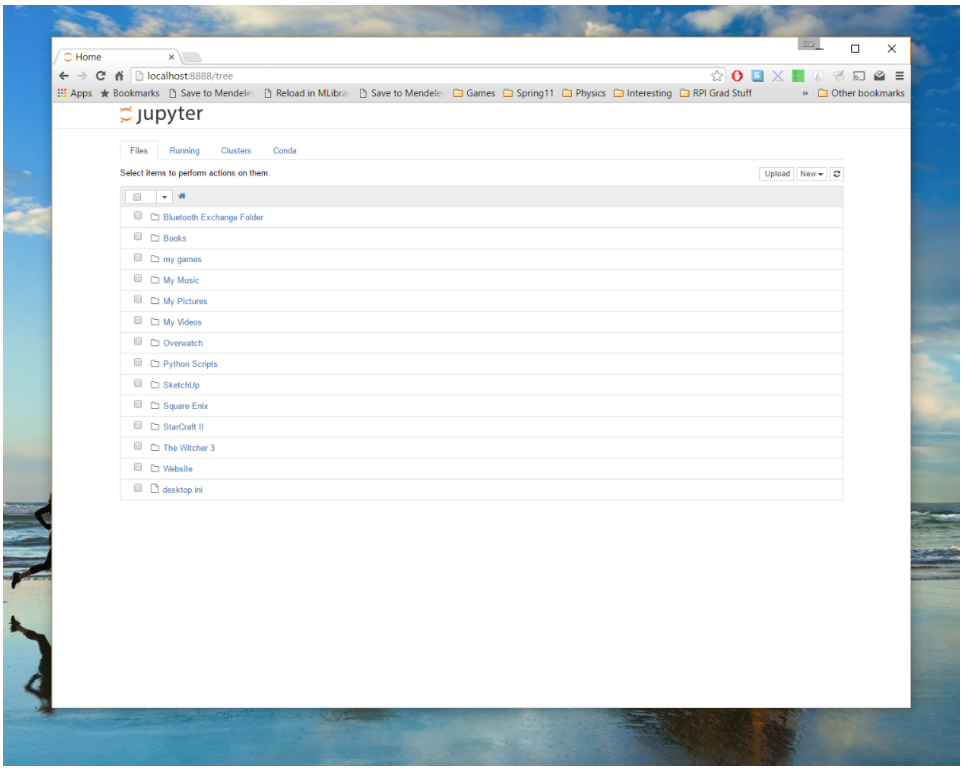


Figure 1: Jupyter on Windows in a new tab in Chrome

and double click the file. This will open up another new tab containing the document, and it is here that you can edit and run the code.

Once you have the tutorial notebook open, follow along with the instructions to learn about Python and how to use the Jupyter notebook.

# Help!

This may be a lot of new stuff for you to learn, so you'll probably get confused and stuck at some point. That's okay! You should feel free to ask your friends, the internet, or your instructors for help with any issues. In addition, here are some highly recommended references

- Lectures on Scientific Computing with Pythonby J.R. Johansson. These notes are actually all Jupyter notebooks, and they are a very easy to follow guide for Python, NumPy, and Matplotlib. Only Lectures 0-4 will be relevant for our class.
- Gallery of Interesting IPython Notebooks. This is a super interesting mix of tutorials and application examples. In reading about Jupyter, you'll probably come across lots of references to IPython, since IPython evolved into Jupyter.
- Python Tutorial, straight from the source.
- Markdown Cheatsheetfor writing text in Jupyter notebooks.